Post-Quantum Zero Knowledge, Revisited
(or: How to do Quantum Rewinding Undetectably)

Alex Lombardi
(Simons & Berkeley)

Fermi Ma
(Simons & Berkeley)

Nicholas Spooner
(Warwick)

Big Question:

When are *classical* cryptosystems secure against *quantum* attacks?

(i.e., post-quantum cryptography)

Why aren’t post-quantum assumptions enough?

Misconception: just need to “replace” quantum-broken assumptions (factoring) with post-quantum assumptions (learning with errors).

Why aren’t post-quantum assumptions enough?

Misconception: just need to “replace” quantum-broken assumptions (factoring) with post-quantum assumptions (learning with errors).

security = definition + reduction + assumption

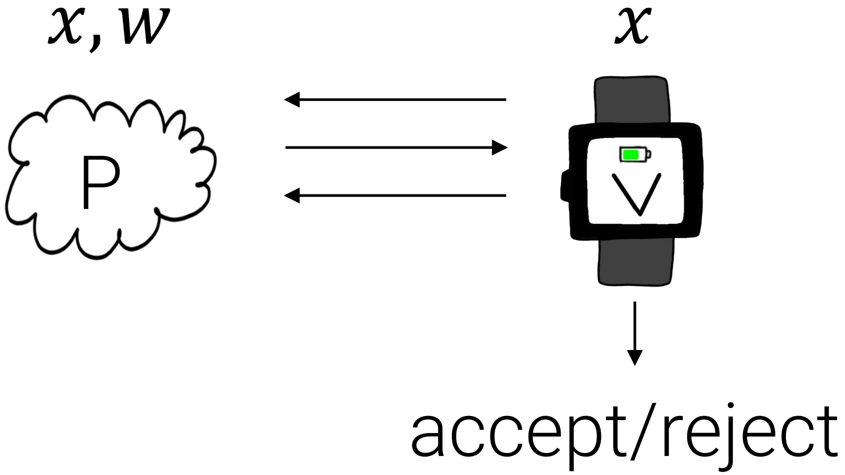
Why aren't post-quantum assumptions enough?

Misconception: just need to "replace" quantum-broken assumptions (factoring) with post-quantum assumptions (learning with errors).

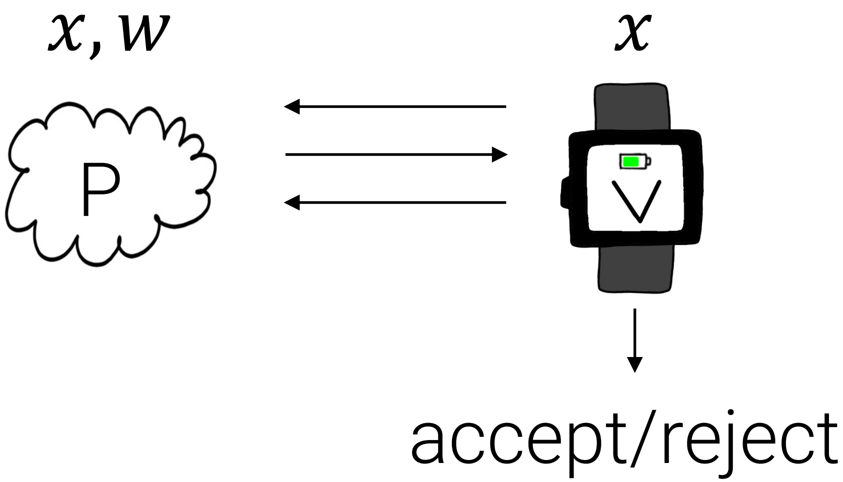
post-quantum security = **post-quantum** definition + **post-quantum** reduction + **post-quantum** assumption

In reality, we also need **definitions** and **reductions** that capture post-quantum security. This can be surprisingly difficult!

This work: zero-knowledge protocols [GMR85]



This work: zero-knowledge protocols [GMR85]



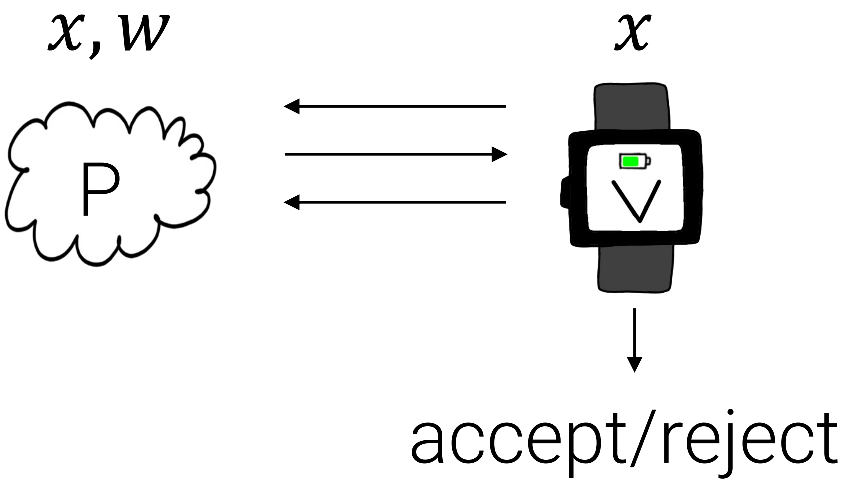
x, w

x

accept/reject

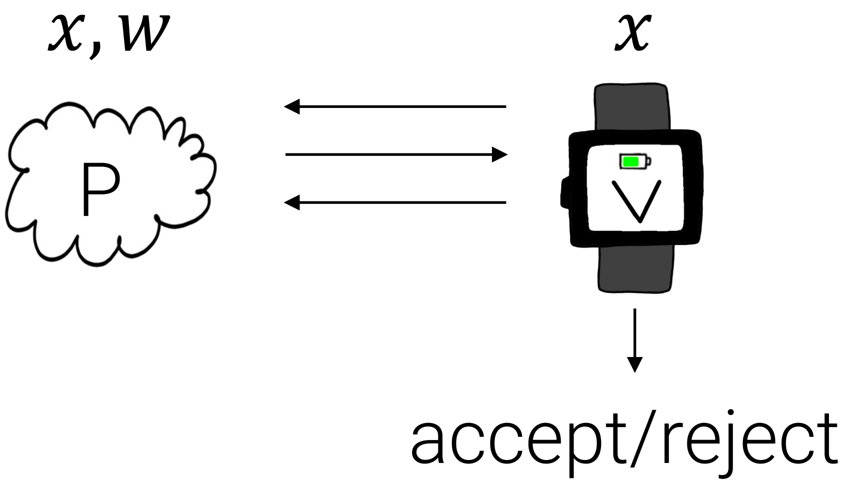
- **Zero knowledge:** proof (of a true statement) can be *efficiently simulated* without the witness.

This work: zero-knowledge protocols [GMR85]



- **Zero knowledge:** proof (of a true statement) can be *efficiently simulated* without the witness.
- **Completeness:** If x is true and P is honest, V accepts.

This work: zero-knowledge protocols [GMR85]



- **Zero knowledge:** proof (of a true statement) can be *efficiently simulated* without the witness.
- **Completeness:** If x is true and P is honest, V accepts.
- **Soundness:** If x is false, P^* cannot make V accept.

We have a deep theory of ZK in the classical setting.

[GMR85,B86,GMW86,GK96,FS90,…]

We have a deep theory of ZK in the classical setting.

[GMR85,B86,GMW86,GK96,FS90,…]

Will this theory hold up against quantum attacks?

We have a deep theory of ZK in the classical setting.

[GMR85,B86,GMW86,GK96,FS90,…]

Will this theory hold up against quantum attacks?

Positive results exist [W07,CCY20,BS20], but a cohesive theory is elusive.

We have a deep theory of ZK in the classical setting.

[GMR85,B86,GMW86,GK96,FS90,…]

Will this theory hold up against quantum attacks?

Positive results exist [W07,CCY20,BS20], but a cohesive theory is elusive. Even many *textbook ZK protocols* are still not understood!

- [Goldreich-Micali-Wigderson86]: graph non-isomorphism
- [Feige-Shamir90]: ZK via “trapdoor extraction”
- [Goldreich-Kahan96]*: five-message proofs for NP

*[CCY21] proved that [GK96] is post-quantum ε-ZK (a weakening of ZK).

This Work

(1) Revisit definition of post-quantum ZK.

This Work

(1) Revisit definition of post-quantum ZK.

- Overcomes a subtle definitional issue highlighted by [CCLY21].

This Work

(1) Revisit definition of post-quantum ZK.

- Overcomes a subtle definitional issue highlighted by [CCLY21].

(2) Extract witness from quantum attacker without disturbing its state.

This Work

(1) Revisit definition of post-quantum ZK.

- Overcomes a subtle definitional issue highlighted by [CCLY21].

(2) Extract witness from quantum attacker without disturbing its state.

Consequences:

- [GMW86] GNI protocol is post-quantum ZK
- [FS90] protocol is post-quantum ZK (*super-poly assumption)

This Work

(1) Revisit definition of post-quantum ZK.

- Overcomes a subtle definitional issue highlighted by [CCLY21].

(2) Extract witness from quantum attacker without disturbing its state.

Consequences:

- [GMW86] GNI protocol is post-quantum ZK
- [FS90] protocol is post-quantum ZK (*super-poly assumption)

See paper: [GK96] protocol for NP is post-quantum ZK.

Plan for Today

1. Recap: GNI protocol + “extract-and-simulate”

2. Challenges in the post-quantum setting

3. This work: defining post-quantum ZK

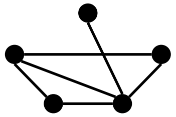
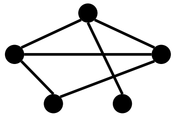
4. This work: quantum extract-and-simulate

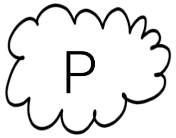
Plan for Today

1. Recap: GNI protocol + “extract-and-simulate”
2. Challenges in the post-quantum setting
3. This work: defining post-quantum ZK
4. This work: quantum extract-and-simulate

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

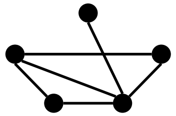
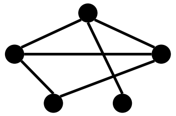
Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

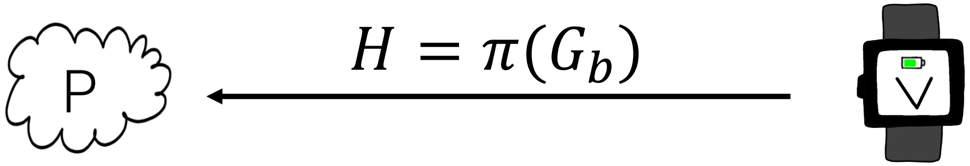
Instance: $G_0 =$  $G_1 =$ 



(simplified protocol)

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 

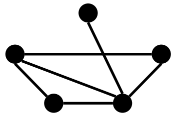
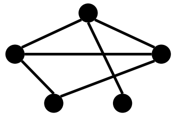


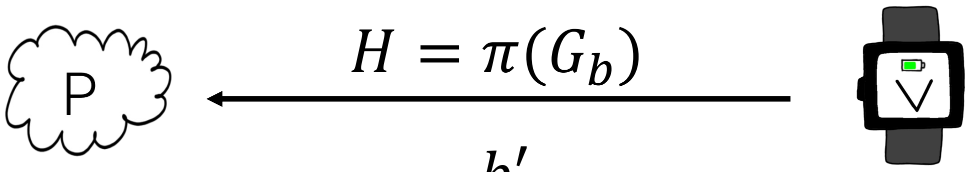
$H = \pi(G_b)$

$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

(simplified protocol)

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



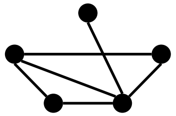
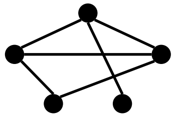
$H = \pi(G_b)$

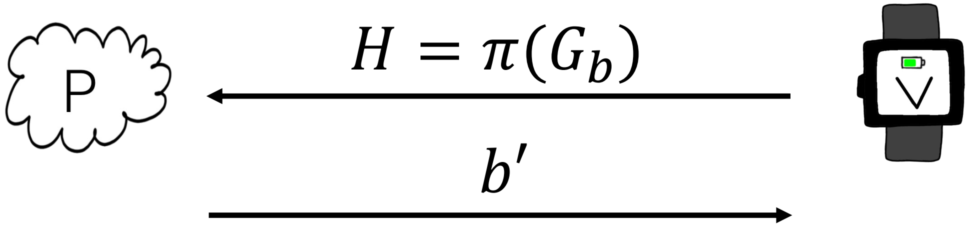
$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

b'

(simplified protocol)

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



$H = \pi(G_b)$

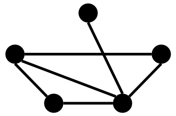
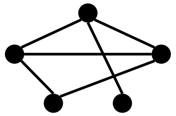
b'

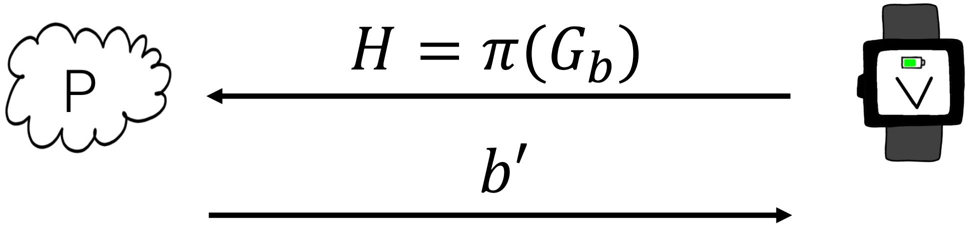
(simplified protocol)

$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

V accepts if $b' = b$.
Repeat for soundness.

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



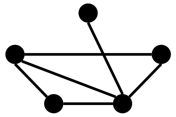
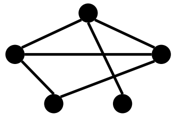
(simplified protocol)

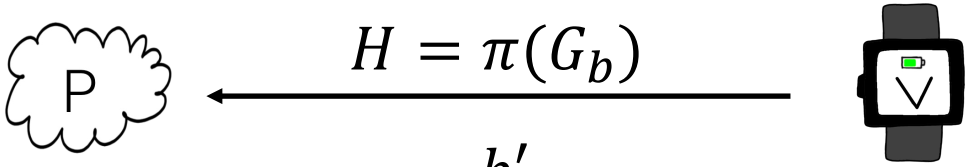
$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

V accepts if $b' = b$.
Repeat for soundness.

- P can distinguish $\pi(G_0), \pi(G_1)$ iff graphs are not isomorphic.

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



$H = \pi(G_b)$

b'

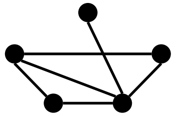
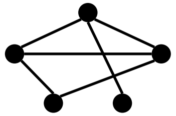
(simplified protocol)

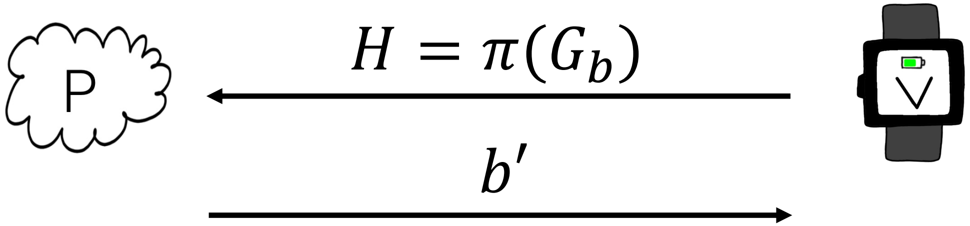
$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

V accepts if $b' = b$.
Repeat for soundness.

- P can distinguish $\pi(G_0), \pi(G_1)$ iff graphs are not isomorphic.
- ZK holds against honest verifiers, since honest verifier already knows b.

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



$H = \pi(G_b)$

b'

(simplified protocol)

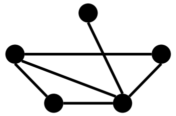
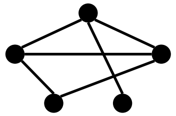
$b \leftarrow \{0,1\}, \pi \leftarrow S_n$.

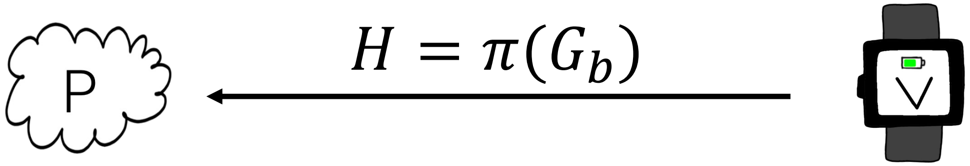
V accepts if $b' = b$.
Repeat for soundness.

- P can distinguish $\pi(G_0), \pi(G_1)$ iff graphs are not isomorphic.
- ZK holds against honest verifiers, since honest verifier already knows b.

In the full [GMW86] protocol, V *proves that it knows b* before P sends b'.

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 

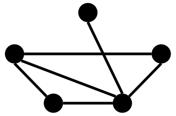
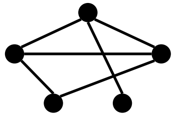


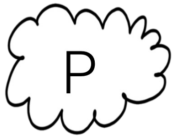
$H = \pi(G_b)$

$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

[GMW86] Proof-of-Knowledge Subprotocol: V proves it “knows” b.

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



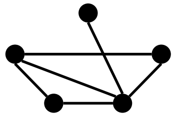
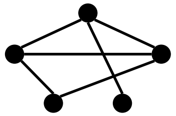
$H = \pi(G_b)$ ←

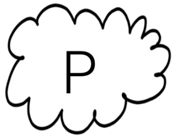
$b \leftarrow \{0,1\}, \pi \leftarrow S_n$.

[GMW86] Proof-of-Knowledge Subprotocol: V proves it "knows" b.

For $i \in [m]$: $G_{i,0}, G_{i,1}$ ← randomly permute (G_0, G_1) or (G_1, G_0)

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



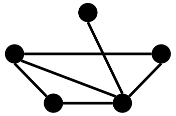
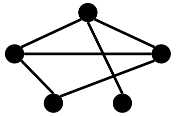
$H = \pi(G_b)$

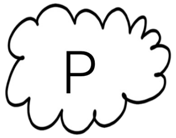
$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

[GMW86] Proof-of-Knowledge Subprotocol: V proves it “knows” b.

For $i \in [m]$: $G_{i,0}, G_{i,1}$ randomly permute (G_0, G_1) or (G_1, G_0)

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



$H = \pi(G_b)$ ←

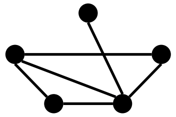
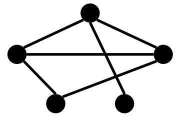
$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

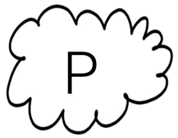
[GMW86] Proof-of-Knowledge Subprotocol: V proves it "knows" b.

For $i \in [m]$: ← $G_{i,0}, G_{i,1}$ randomly permute (G_0, G_1) or (G_1, G_0)

$r_i \leftarrow \{0,1\}$ → r_i

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



$\xleftarrow{H = \pi(G_b)}$

$b \leftarrow \{0,1\}, \pi \leftarrow S_n$.

[GMW86] Proof-of-Knowledge Subprotocol: V proves it "knows" b.

For $i \in [m]$: $\xleftarrow{G_{i,0}, G_{i,1}}$ randomly permute (G_0, G_1) or (G_1, G_0)

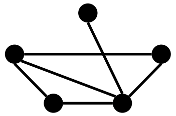
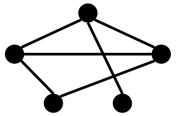
$r_i \leftarrow \{0,1\}$ $\xrightarrow{r_i}$

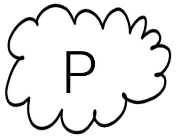
$\xleftarrow{z_i}$

If $r_i = 0$, reveal both permutations

If $r_i = 1$, reveal permutation between H and one of $G_{i,0}, G_{i,1}$

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



$H = \pi(G_b)$ ←

$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

[GMW86] Proof-of-Knowledge Subprotocol: V proves it "knows" b.

For $i \in [m]$:

← $G_{i,0}, G_{i,1}$ randomly permute (G_0, G_1) or (G_1, G_0)

$r_i \leftarrow \{0,1\}$ r_i →

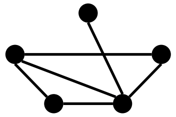
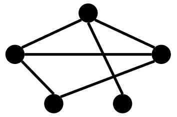
← z_i

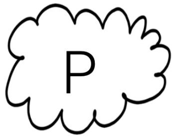
If $r_i = 0$, reveal both permutations

If $r_i = 1$, reveal permutation between H and one of $G_{i,0}, G_{i,1}$

Why does this prove knowledge of b? If V^* can answer both challenges correctly, it must know which of G_0, G_1 is isomorphic to H

Recall: [GMW86] ZK Protocol for Graph Non-Isomorphism

Instance: $G_0 =$  $G_1 =$ 



$\xleftarrow{H = \pi(G_b)}$

$b \leftarrow \{0,1\}, \pi \leftarrow S_n.$

[GMW86] Proof-of-Knowledge Subprotocol: V proves it "knows" b.

For $i \in [m]$: $\xleftarrow{G_{i,0}, G_{i,1}}$ randomly permute (G_0, G_1) or (G_1, G_0)

$r_i \leftarrow \{0,1\}$ $\xrightarrow{r_i}$

$\xleftarrow{z_i}$

If $r_i = 0$, reveal both permutations

If $r_i = 1$, reveal permutation between H and one of $G_{i,0}, G_{i,1}$

$\xrightarrow{b'}$

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.

Instance:

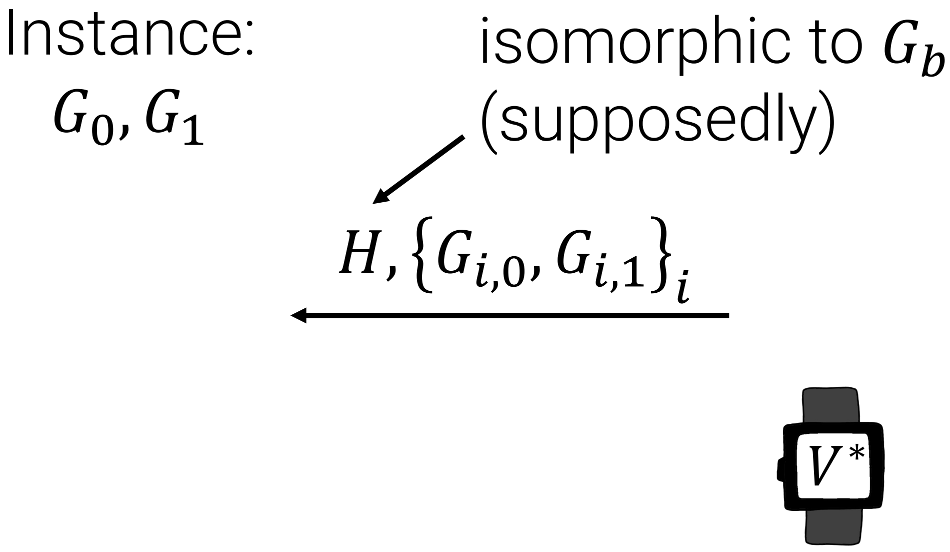
G_0, G_1

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.



Instance:
G_0, G_1

isomorphic to G_b
(supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

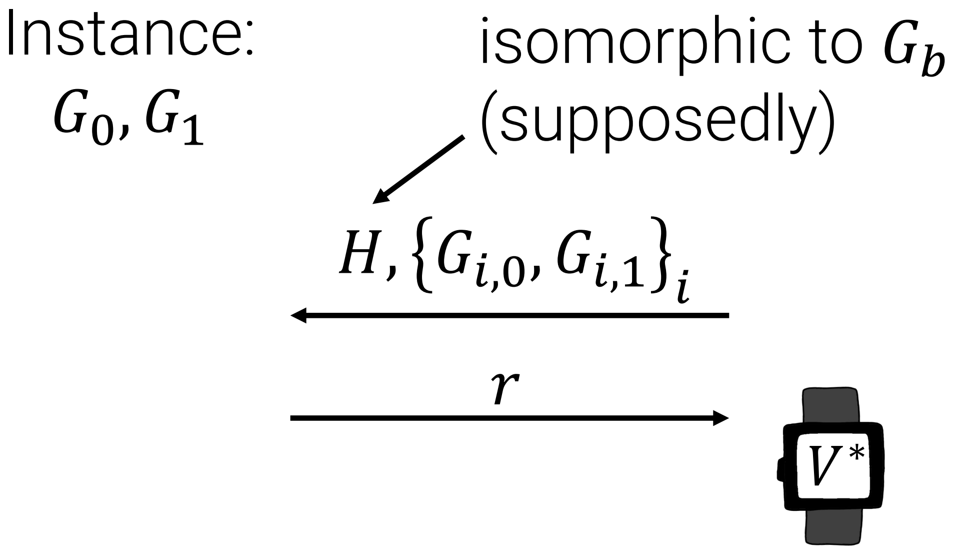
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.



Instance:

G_0, G_1

isomorphic to G_b (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

r

V^*

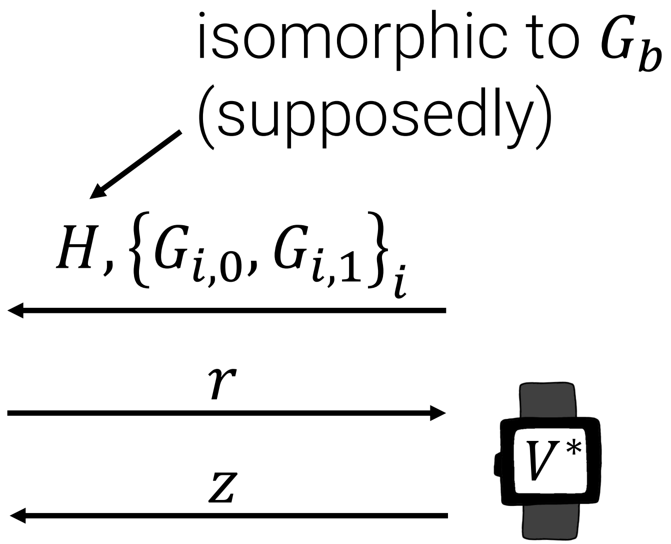
Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.

Instance: G_0, G_1



isomorphic to G_b (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

r

z

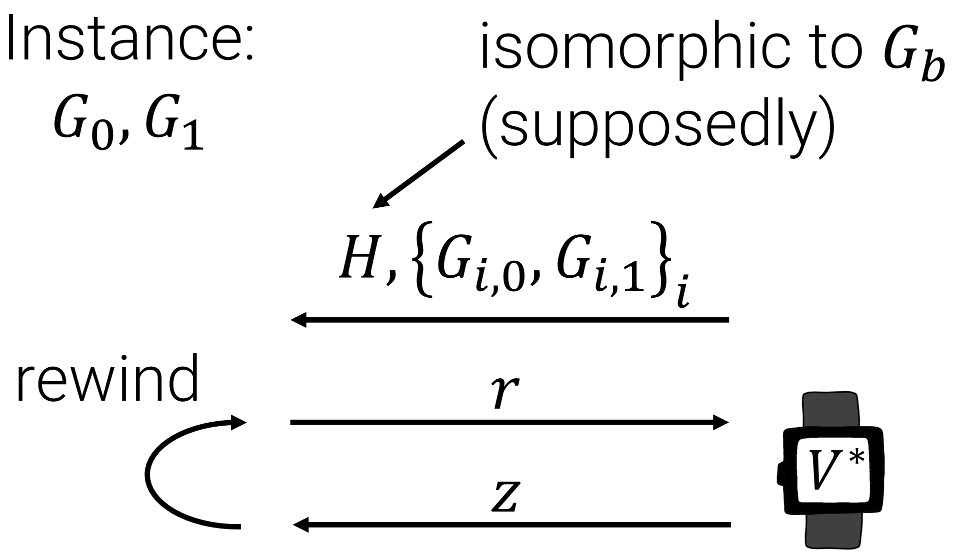
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.



Instance:
G_0, G_1

isomorphic to G_b
(supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

r

z

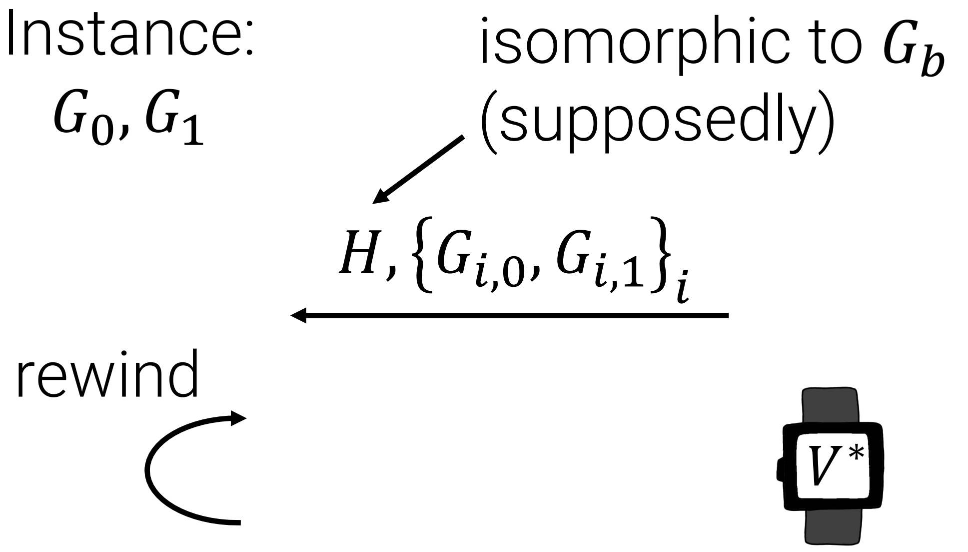
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.



Instance: G_0, G_1

isomorphic to G_b (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

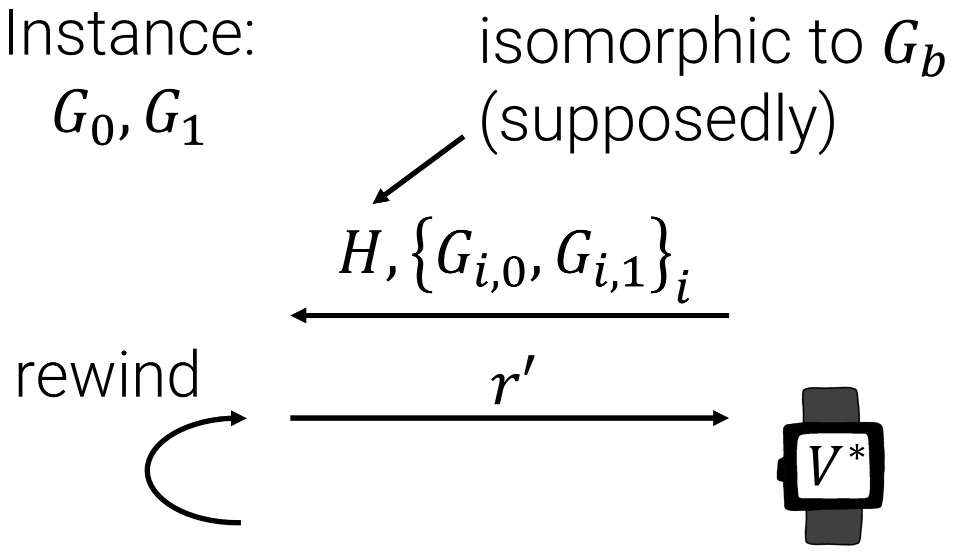
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.



Instance:
G_0, G_1

isomorphic to G_b (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

r'

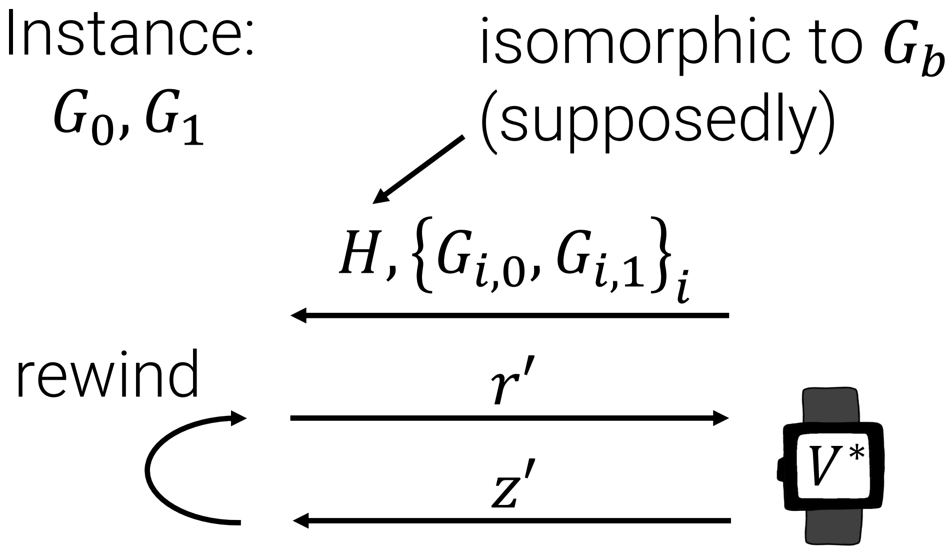
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.



Instance:
G_0, G_1

isomorphic to G_b (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

r'

z'

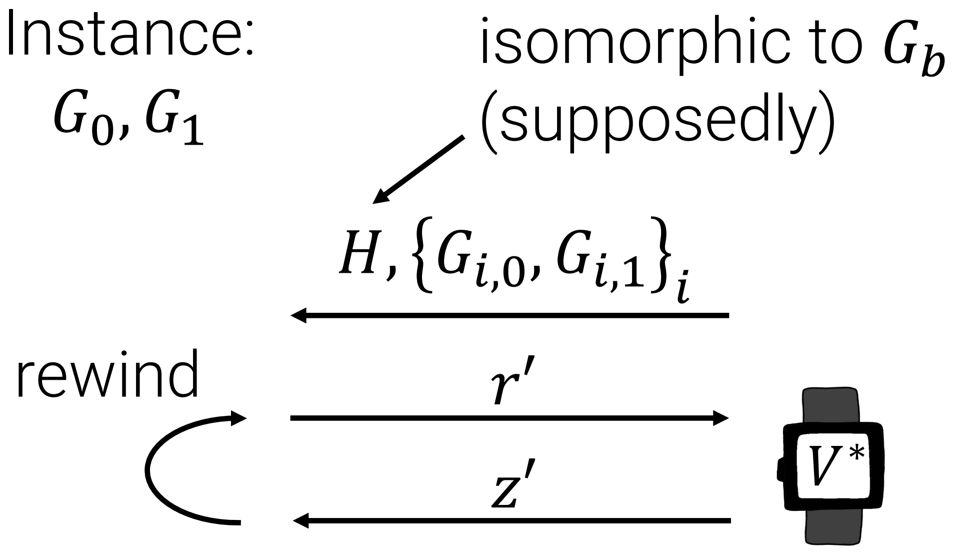
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract b.



Instance:
G_0, G_1

isomorphic to G_b
(supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

r'

z'

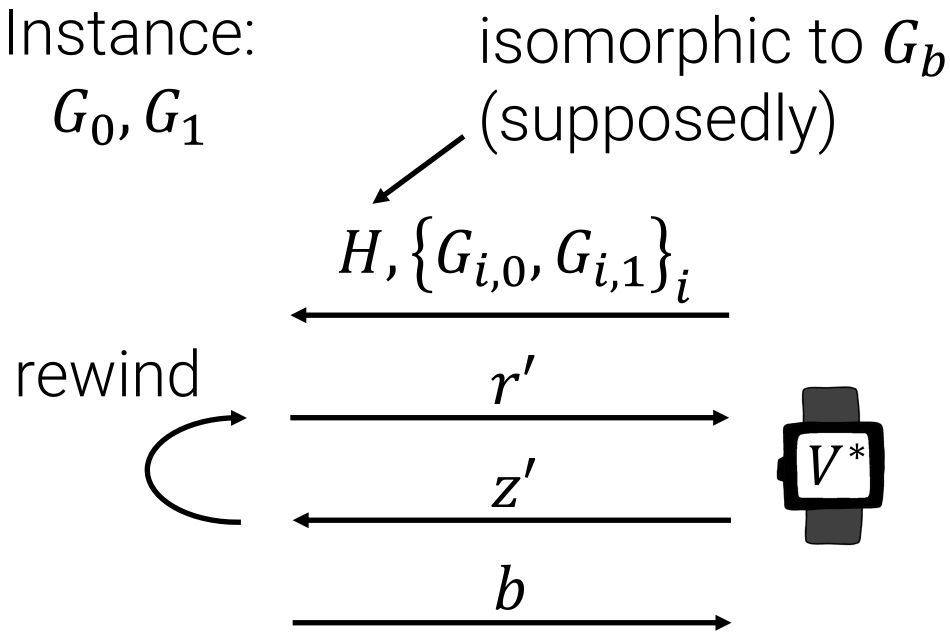
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract b.



Instance:
G_0, G_1

isomorphic to G_b
(supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

r'

z'

b

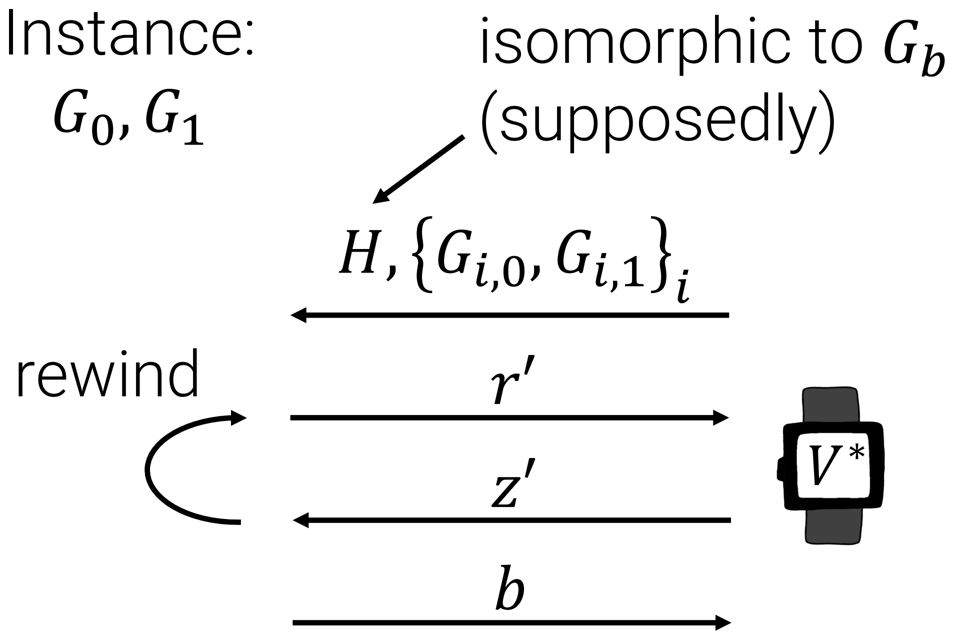
V^*

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract b.



Instance: G_0, G_1

isomorphic to G_b (supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

r'

z'

b

V^*

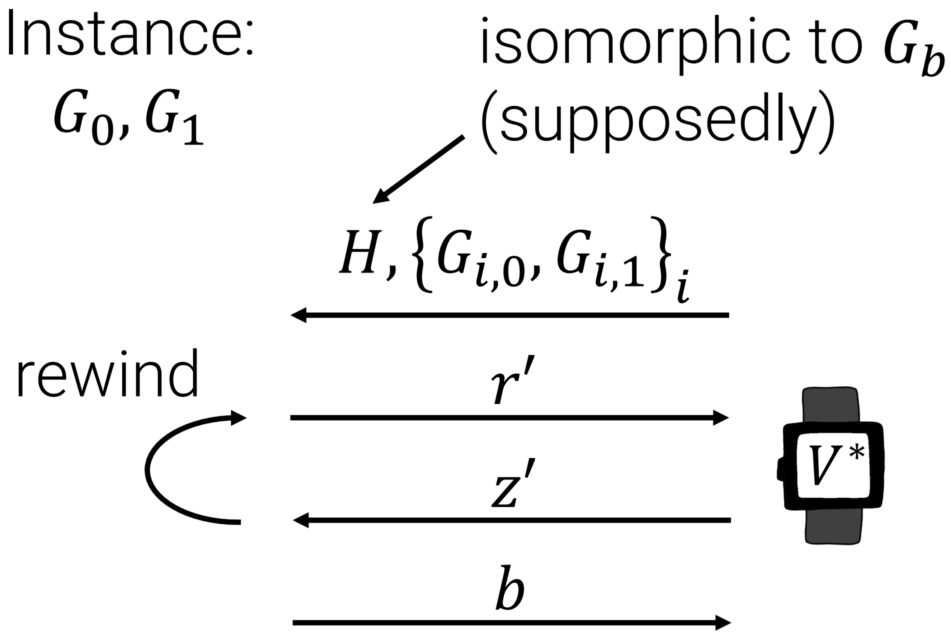
This "extract and simulate" approach appears in many textbook ZK protocols [GMR85,GMW86,FS90]

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract b.



Instance:
G_0, G_1

isomorphic to G_b
(supposedly)

$H, \{G_{i,0}, G_{i,1}\}_i$

rewind

r'

z'

b

V^*

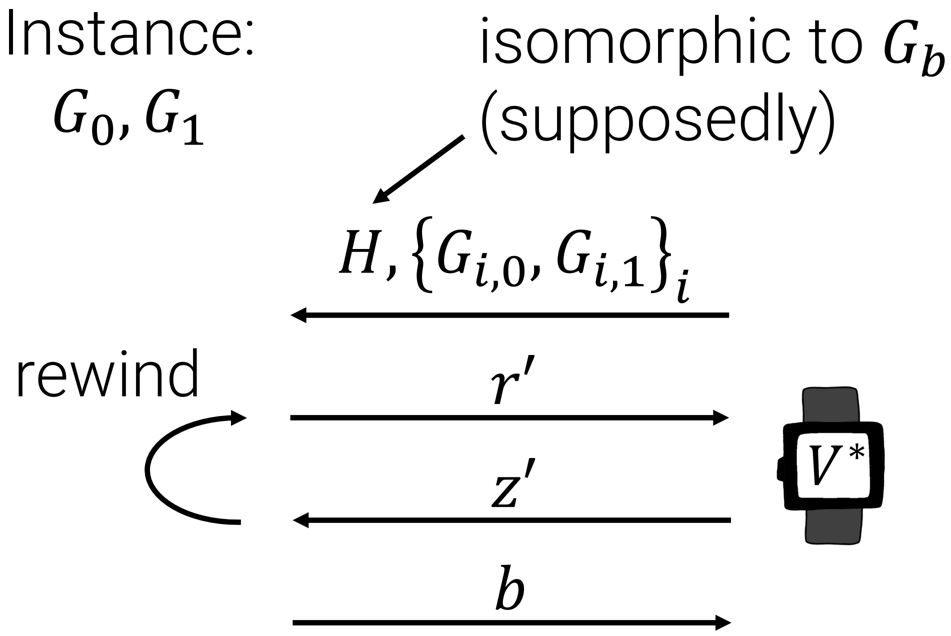
Important point: what is the simulator's runtime?

Extract-and-Simulate Paradigm

ZK requires a simulator that outputs b if V^* outputs accepting z.

[GMW86] ZK Simulator

1) Run V^* once, check if z is accepting. If not, done.
2) Rewind V^* and query it until it outputs a second accepting z'.
3) Given two accepting transcripts $(r, z), (r', z')$ where $r \neq r'$, simulator can extract b.



Important point: what is the simulator's runtime?

ZK def allows *expected poly-time* simulation $(\varepsilon \cdot \frac{1}{\varepsilon} = 1)$

Plan for Today

1. Recap: GNI protocol + “extract-and-simulate” ✓
2. Challenges in the post-quantum setting
3. This work: defining post-quantum ZK
4. This work: quantum extract-and-simulate

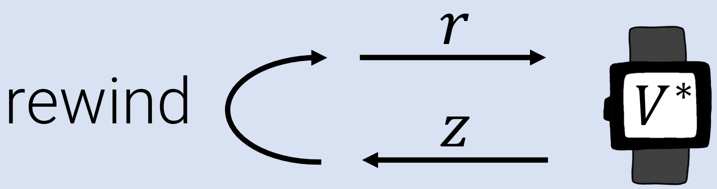
For post-quantum ZK, we need to:

(1) Extract b from quantum V^*.

For post-quantum ZK, we need to:

(1) Extract b from quantum V^*.

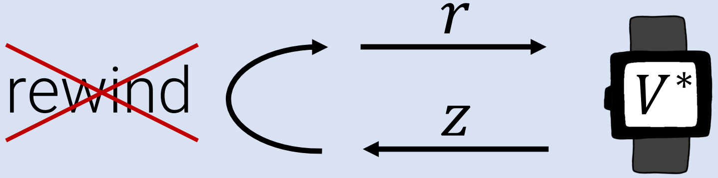
- Classically sim records z and rewinds V^* to an earlier state.



For post-quantum ZK, we need to:

(1) Extract b from quantum V^*.

- Classically sim records z and rewinds V^* to an earlier state.

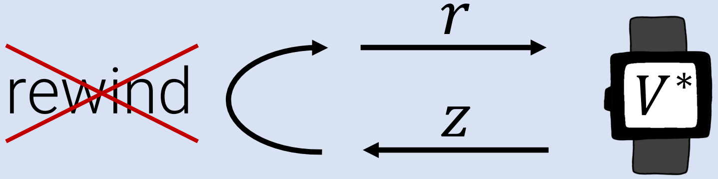


- If V^* is quantum, recording z can disturb V^*'s state!

For post-quantum ZK, we need to:

(1) Extract b from quantum V^*.

- Classically sim records z and rewinds V^* to an earlier state.



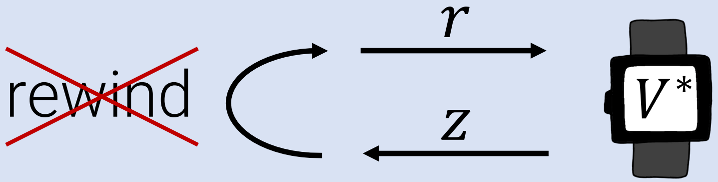
- If V^* is quantum, recording z can disturb V^*'s state!

(2) Simulate internal quantum state of V^*.

For post-quantum ZK, we need to:

(1) Extract b from quantum V^*.

- Classically sim records z and rewinds V^* to an earlier state.



- If V^* is quantum, recording z can disturb V^*'s state!

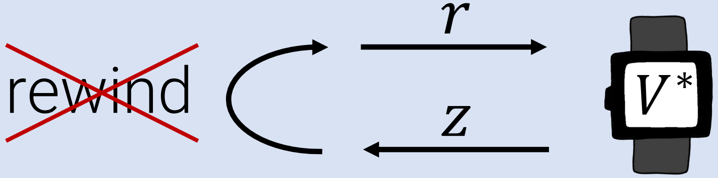
(2) Simulate internal quantum state of V^*.

- [U12,CMSZ21] can handle (1), but will disturb the state.

For post-quantum ZK, we need to:

(1) Extract b from quantum V^*.

- Classically sim records z and rewinds V^* to an earlier state.



- If V^* is quantum, recording z can disturb V^*'s state!

(2) Simulate internal quantum state of V^*.

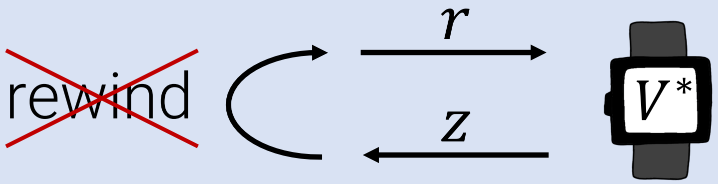
- [U12,CMSZ21] can handle (1), but will disturb the state.

(3) Run in expected quantum poly time.

For post-quantum ZK, we need to:

(1) Extract b from quantum V^*.

- Classically sim records z and rewinds V^* to an earlier state.



- If V^* is quantum, recording z can disturb V^*'s state!

(2) Simulate internal quantum state of V^*.

- [U12,CMSZ21] can handle (1), but will disturb the state.

(3) Run in expected quantum poly time.

- Defining variable-time quantum computation is subtle. [M97,O98,LP98]

Plan for Today

1. Recap: GNI protocol + “extract-and-simulate” ✓
2. Challenges in the post-quantum setting ✓
3. This work: defining post-quantum ZK
4. This work: quantum extract-and-simulate

How should post-quantum ZK even be defined?

(for expected poly-time simulation)

How should post-quantum ZK even be defined?

(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any classical poly-time V^* can be simulated in classical ***expected*** poly time.*

How should post-quantum ZK even be defined?

(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any classical poly-time V^* can be simulated in classical ***expected*** poly time.*

[BL02]: Classical ***strict*** poly-time ZK sim is impossible (black-box, constant-round).

How should post-quantum ZK even be defined?

(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any classical poly-time V^* can be simulated in classical ***expected*** poly time.*

Obvious (?) post-quantum ZK def
View of any quantum poly-time V^* can be simulated in quantum ***expected*** poly time.

[BL02]: Classical ***strict*** poly-time ZK sim is impossible (black-box, constant-round).

How should post-quantum ZK even be defined?

(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any classical poly-time V^* can be simulated in classical ***expected*** poly time.*

Obvious (?) post-quantum ZK def
View of any quantum poly-time V^* can be simulated in quantum ***expected*** poly time.

[BL02]: Classical ***strict*** poly-time ZK sim is impossible (black-box, constant-round).

[CCLY21]: Quantum ***expected*** poly-time ZK sim is impossible* (black-box, constant-round).

How should post-quantum ZK even be defined?

(for expected poly-time simulation)

Original ZK def [GMR85]:
View of any classical poly-time V^* can be simulated in classical ***expected*** poly time.*

Obvious (?) post-quantum ZK def
View of any quantum poly-time V^* can be simulated in quantum ***expected*** poly time.

[BL02]: Classical ***strict*** poly-time ZK sim is impossible (black-box, constant-round).

[CCLY21]: Quantum ***expected*** poly-time ZK sim is impossible* (black-box, constant-round).

This work: new ZK def that circumvents CCLY21 barrier.

This Work: Measured vs. Coherent EQPT Simulation

This Work: Measured vs. Coherent EQPT Simulation

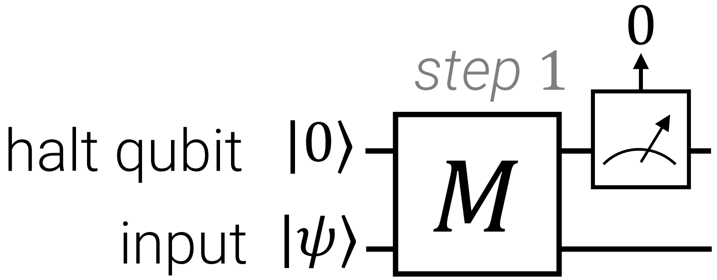
Measured EQPT

halt qubit $|0\rangle$ —

input $|\psi\rangle$ —

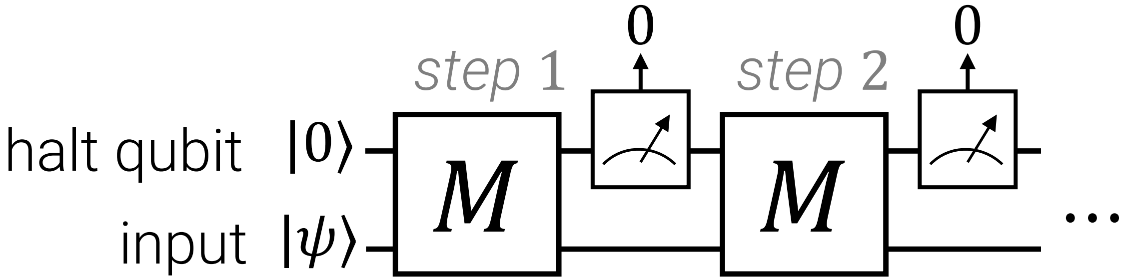
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT



This Work: Measured vs. Coherent EQPT Simulation

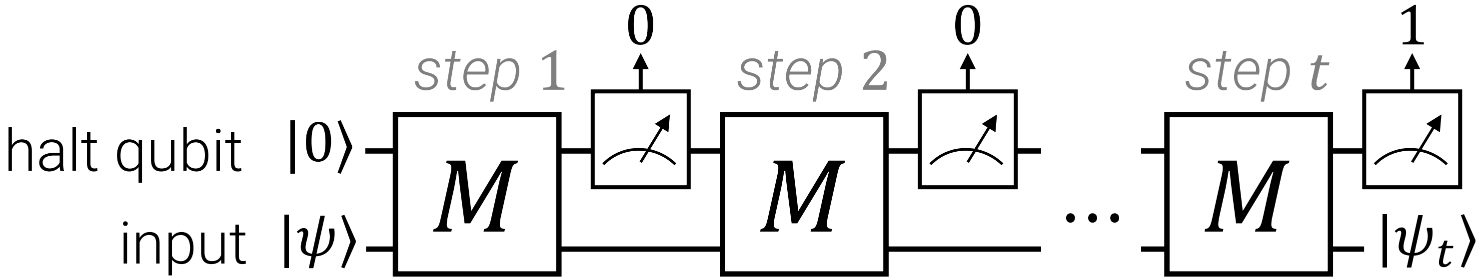
Measured EQPT



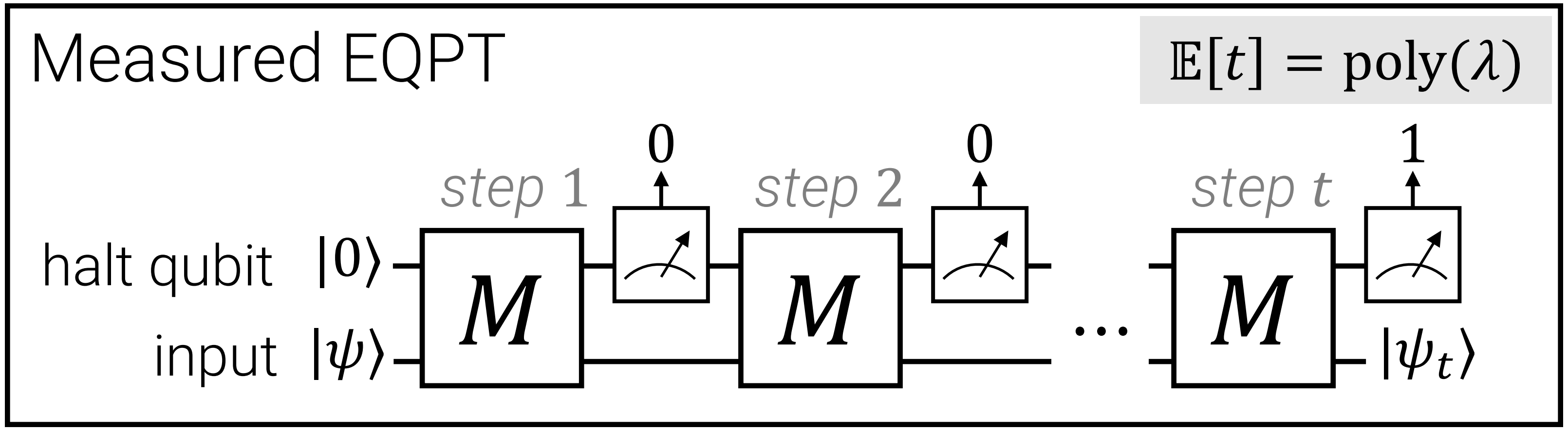
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

$\mathbb{E}[t] = \mathrm{poly}(\lambda)$



This Work: Measured vs. Coherent EQPT Simulation

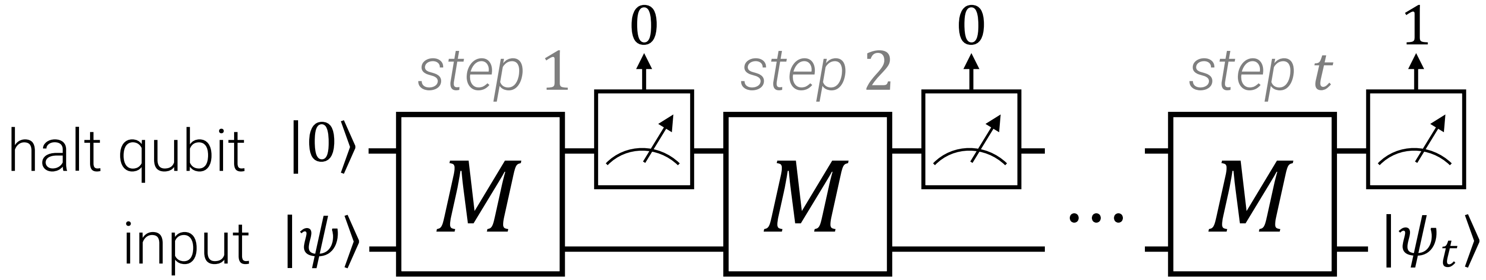


[CCLY21]: measuring simulator's runtime disturbs verifier's state.

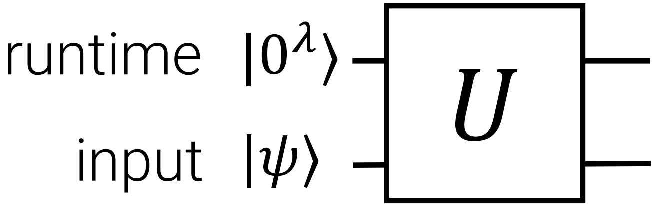
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

$\mathbb{E}[t] = \text{poly}(\lambda)$



Coherent EQPT



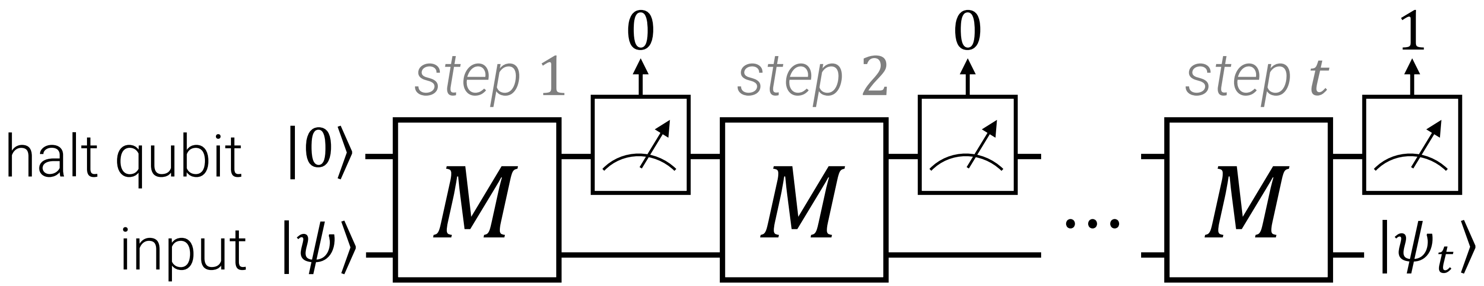
U runs a measured-EQPT procedure coherently for 2^λ steps, leaving t *in superposition*.

[CCLY21]: measuring simulator's runtime disturbs verifier's state.

This Work: Measured vs. Coherent EQPT Simulation

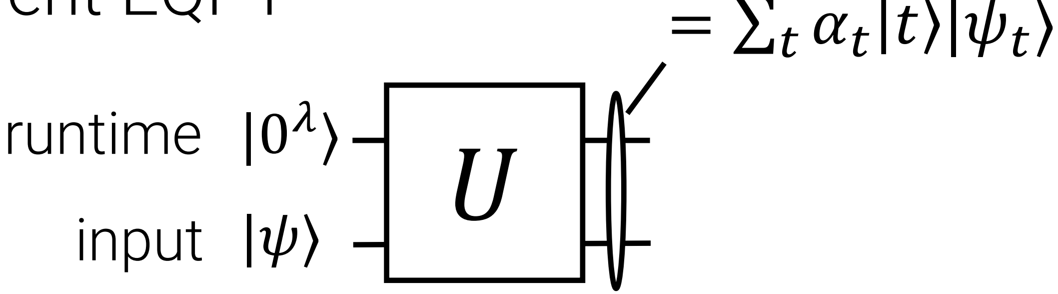
Measured EQPT

$\mathbb{E}[t] = \mathrm{poly}(\lambda)$



[CCLY21]: measuring simulator's runtime disturbs verifier's state.

Coherent EQPT



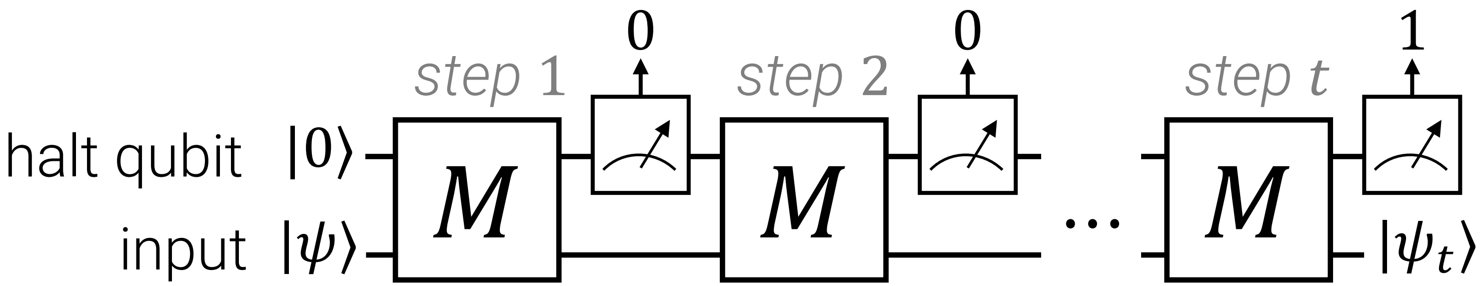
$= \sum_t \alpha_t |t\rangle|\psi_t\rangle$

U runs a measured-EQPT procedure coherently for 2^λ steps, leaving t *in superposition*.

This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

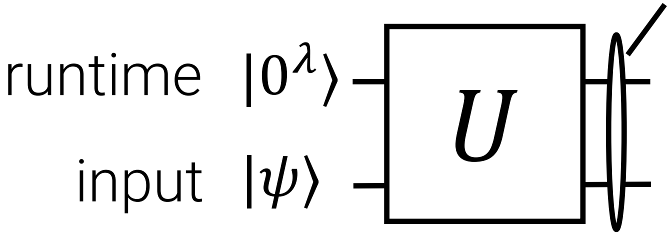
$\mathbb{E}[t] = \mathrm{poly}(\lambda)$



[CCLY21]: measuring simulator's runtime disturbs verifier's state.

Coherent EQPT

$= \sum_t \alpha_t |t\rangle|\psi_t\rangle$



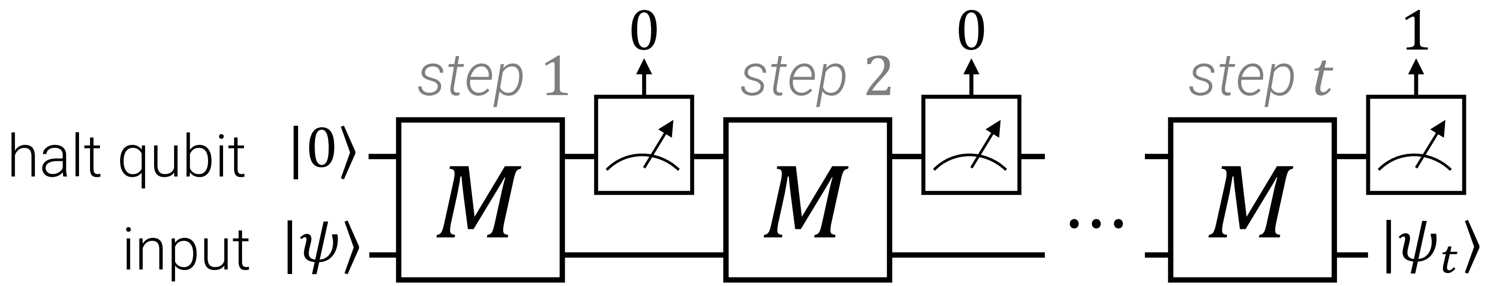
If we measure t here, $\mathbb{E}[t] = \mathrm{poly}(\lambda)$

U runs a measured-EQPT procedure coherently for 2^λ steps, leaving t *in superposition*.

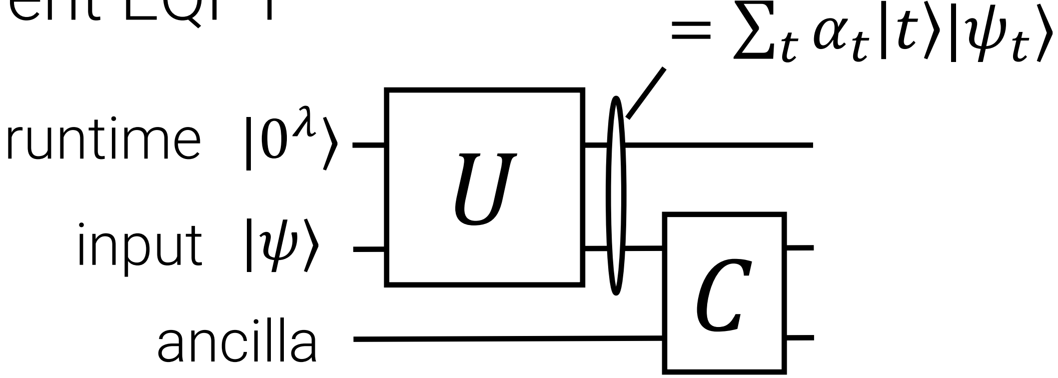
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

$\mathbb{E}[t] = \mathrm{poly}(\lambda)$



Coherent EQPT



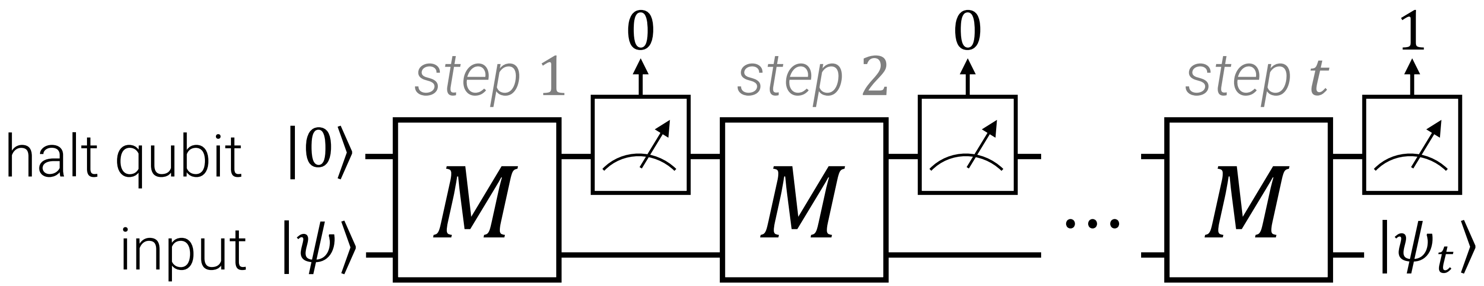
U runs a measured-EQPT procedure coherently for 2^λ steps, leaving t *in superposition*. C is a unitary (e.g., SWAP).

[CCLY21]: measuring simulator's runtime disturbs verifier's state.

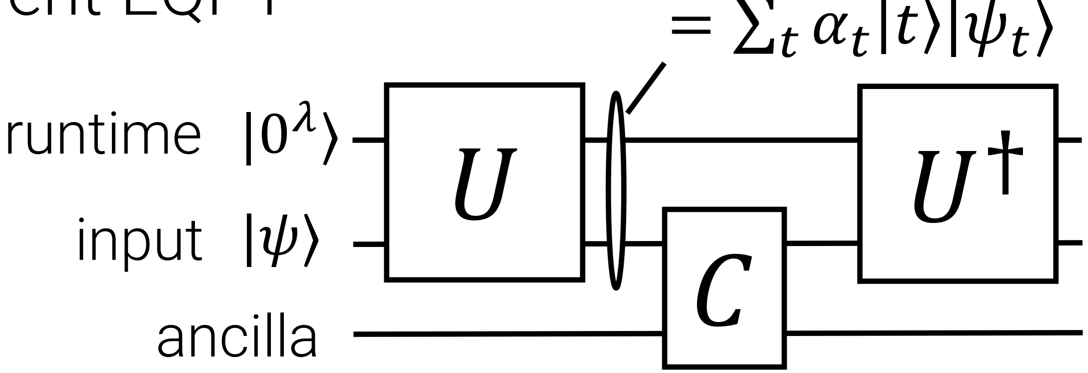
This Work: Measured vs. Coherent EQPT Simulation

Measured EQPT

$\mathbb{E}[t] = \mathrm{poly}(\lambda)$



Coherent EQPT



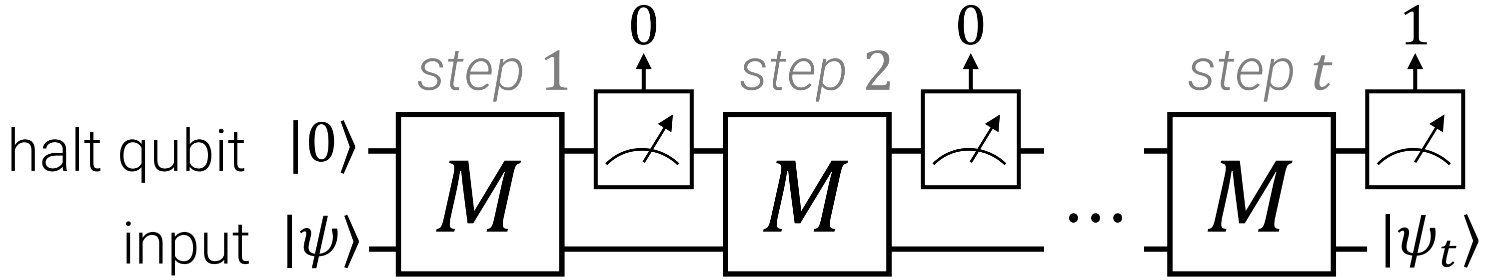
U runs a measured-EQPT procedure coherently for 2^λ steps, leaving t *in superposition*. C is a unitary (e.g., SWAP).

[CCLY21]: measuring simulator's runtime disturbs verifier's state.

This Work: Measured vs. Coherent EQPT Simulation

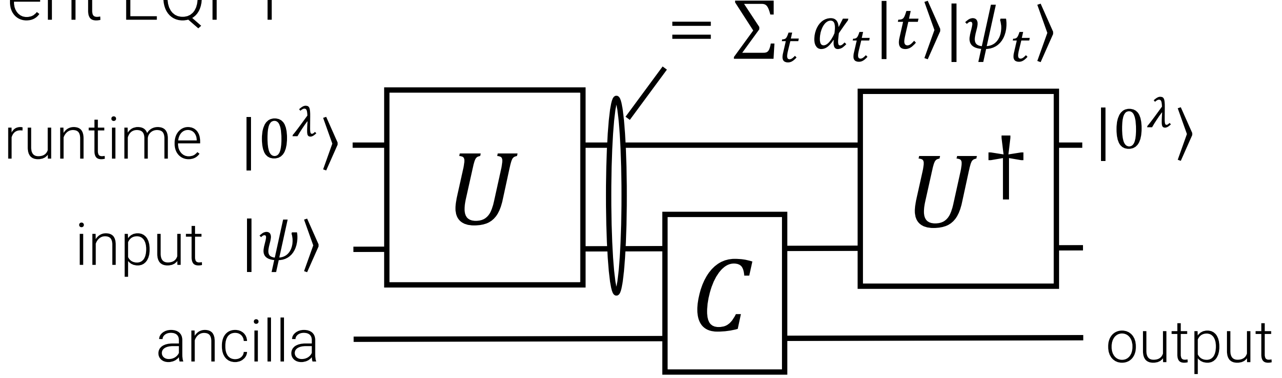
Measured EQPT

$\mathbb{E}[t] = \mathrm{poly}(\lambda)$



[CCLY21]: measuring simulator's runtime disturbs verifier's state.

Coherent EQPT



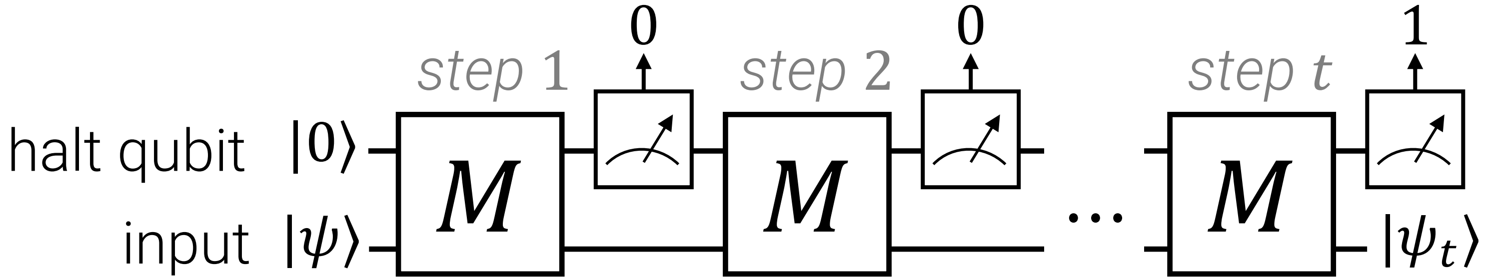
U runs a measured-EQPT procedure coherently for 2^λ steps, leaving t *in superposition*. C is a unitary (e.g., SWAP).

Coherent EQPT leaves runtime in superposition and later *uncomputes* it.

This Work: Measured vs. Coherent EQPT Simulation

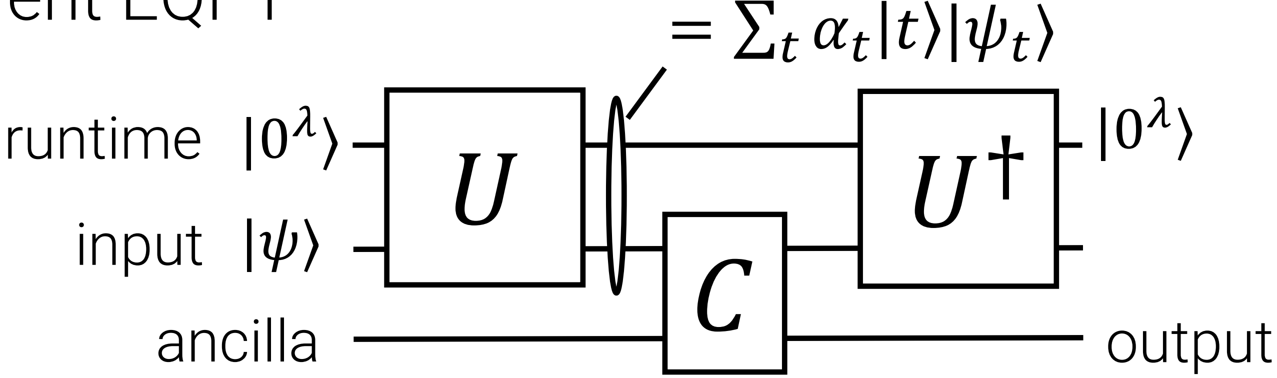
Measured EQPT

$\mathbb{E}[t] = \mathrm{poly}(\lambda)$



[CCLY21]: measuring simulator’s runtime disturbs verifier’s state.

Coherent EQPT



U runs a measured-EQPT procedure coherently for 2^λ steps, leaving t *in superposition*. C is a unitary (e.g., SWAP).

Coherent EQPT leaves runtime in superposition and later *uncomputes* it.

CCLY21 does not rule out coherent EQPT simulation!

Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

Related: why is EPT simulation reasonable in classical ZK?

Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

Related: why is EPT simulation reasonable in classical ZK?

1) **EPT is no stronger than poly-time.**

Ex: if assumption A is broken in EPT, it's also broken in poly time.

Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

Related: why is EPT simulation reasonable in classical ZK?

1) **EPT is no stronger than poly-time.**

Ex: if assumption A is broken in EPT, it's also broken in poly time.

2) **EPT appears to be the weakest model that makes ZK possible.***

[BL02]: poly-time sim is impossible.*

*for constant-round protocols
w/ black-box simulation

Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

We show an analogous state of affairs for coherent EQPT.

Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

We show an analogous state of affairs for coherent EQPT.

1) Coherent EQPT is no stronger than QPT.

Ex: if assumption A is broken in coherent EQPT, it’s also broken in QPT.

Why is it reasonable to define post-quantum ZK with coherent EQPT simulation?

We show an analogous state of affairs for coherent EQPT.

1) Coherent EQPT is no stronger than QPT.

Ex: if assumption A is broken in coherent EQPT, it's also broken in QPT.

2) Coherent EQPT appears to be the weakest model that makes post-quantum ZK possible.*

[CCLY21]: measured EQPT sim impossible.*

*for constant-round protocols
w/ black-box simulation

Plan for Today

1. Recap: GNI protocol + “extract-and-simulate” ✓
2. Challenges in the post-quantum setting ✓
3. This work: defining post-quantum ZK ✓
4. This work: quantum extract-and-simulate

Goal: build coherent EQPT simulator that extracts b from V^* without disturbing its internal state $|\psi\rangle$.

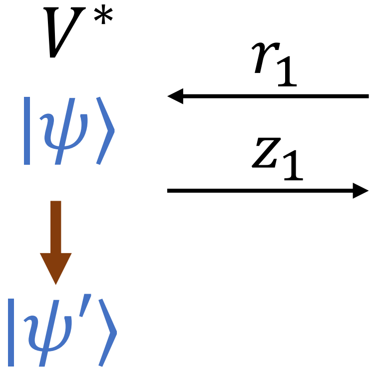
[CMSZ21] allows us to extract b, but will disturb $|\psi\rangle$.

[CMSZ21] allows us to extract b, but will **disturb** $|\psi\rangle$.

CMSZ rewinding: define projector Π onto "useful" verifier states.

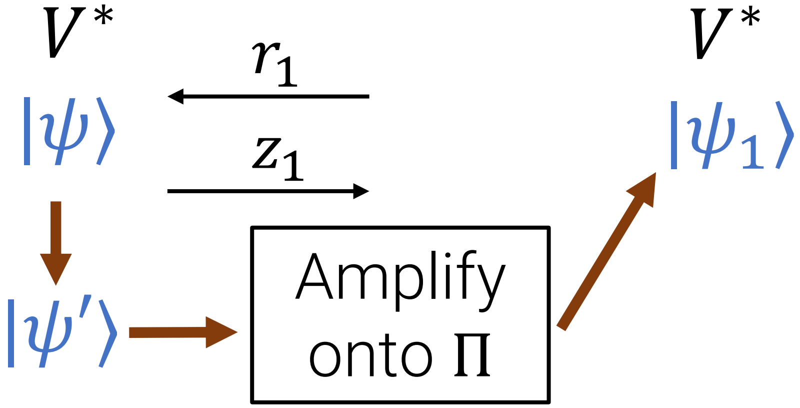
[CMSZ21] allows us to extract b, but will **disturb** $|\psi\rangle$.

CMSZ rewinding: define projector Π onto "useful" verifier states.



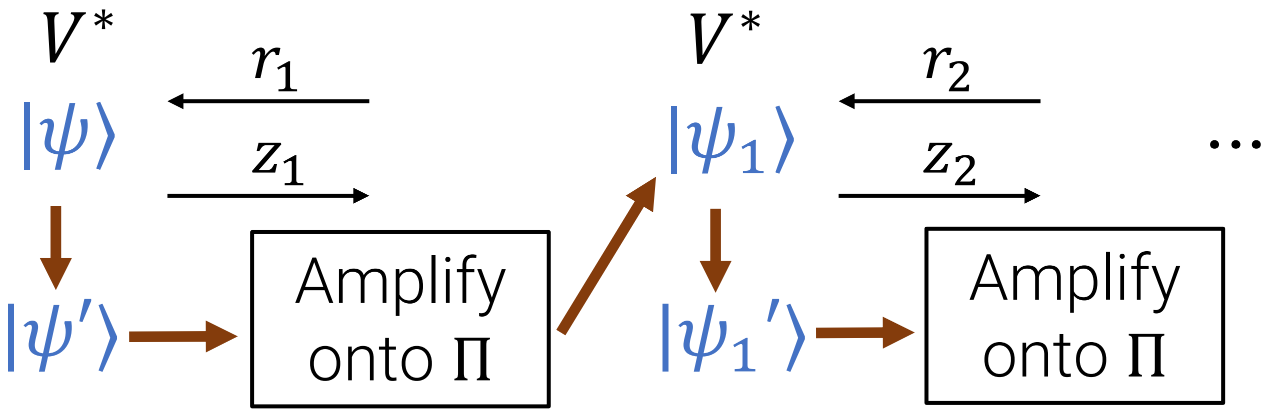
[CMSZ21] allows us to extract b, but will **disturb** $|\psi\rangle$.

CMSZ rewinding: define projector Π onto "useful" verifier states.



[CMSZ21] allows us to extract b, but will **disturb** $|\psi\rangle$.

CMSZ rewinding: define projector Π onto "useful" verifier states.



V^*

$|\psi\rangle$

r_1

z_1

$|\psi'\rangle$

Amplify onto Π

V^*

$|\psi_1\rangle$

r_2

z_2

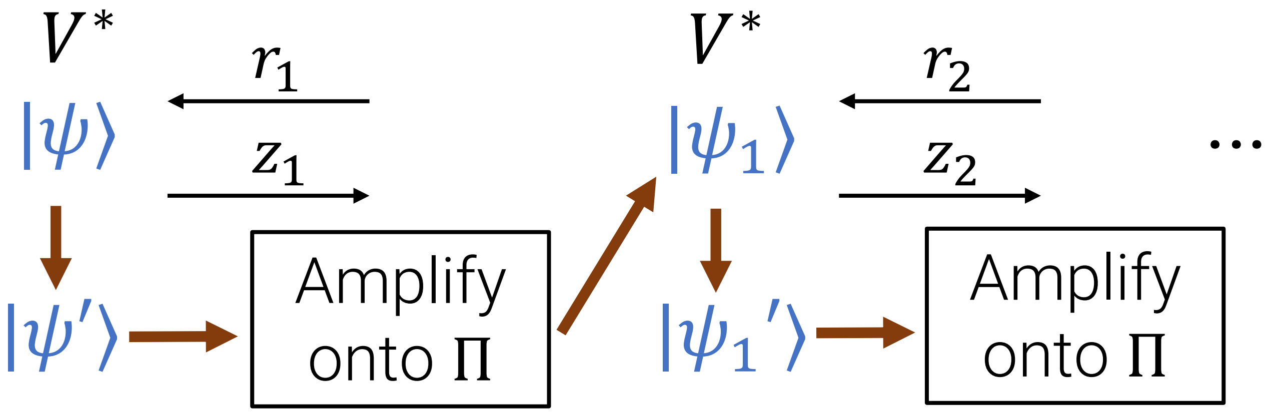
$|\psi_1{}'\rangle$

Amplify onto Π

…

[CMSZ21] allows us to extract b, but will **disturb** $|\psi\rangle$.

CMSZ rewinding: define projector Π onto "useful" verifier states.

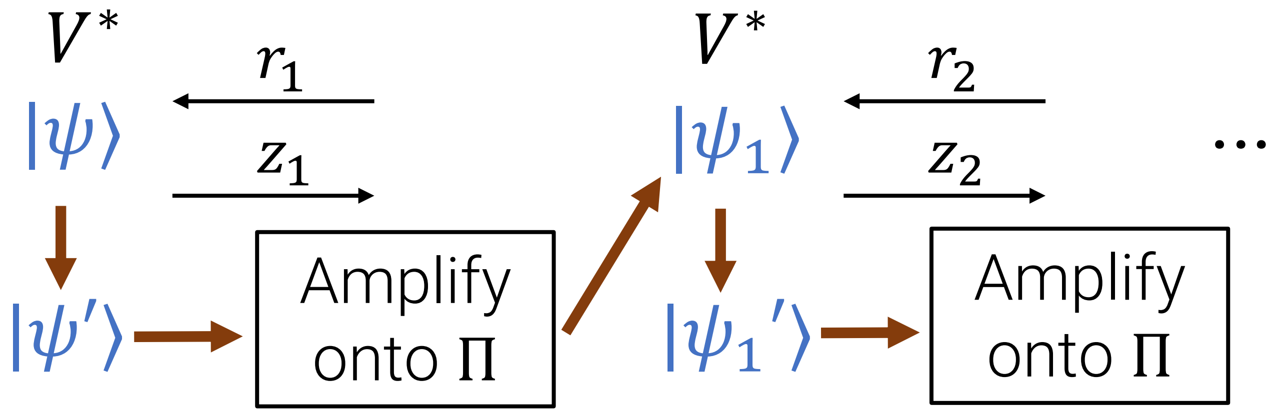


Idea: define variable-time procedure

$\mathrm{Ext} =$"Run CMSZ to get valid $(r, z), (r', z')$"

[CMSZ21] allows us to extract b, but will **disturb** $|\psi\rangle$.

CMSZ rewinding: define projector Π onto "useful" verifier states.



V^*

$|\psi\rangle$

r_1

z_1

$|\psi'\rangle$

Amplify onto Π

V^*

$|\psi_1\rangle$

r_2

z_2

$|\psi_1{}'\rangle$

Amplify onto Π

…

Idea: define variable-time procedure

$\mathrm{Ext} =$"Run CMSZ to get valid $(r,z),(r',z')$"

Simulator:

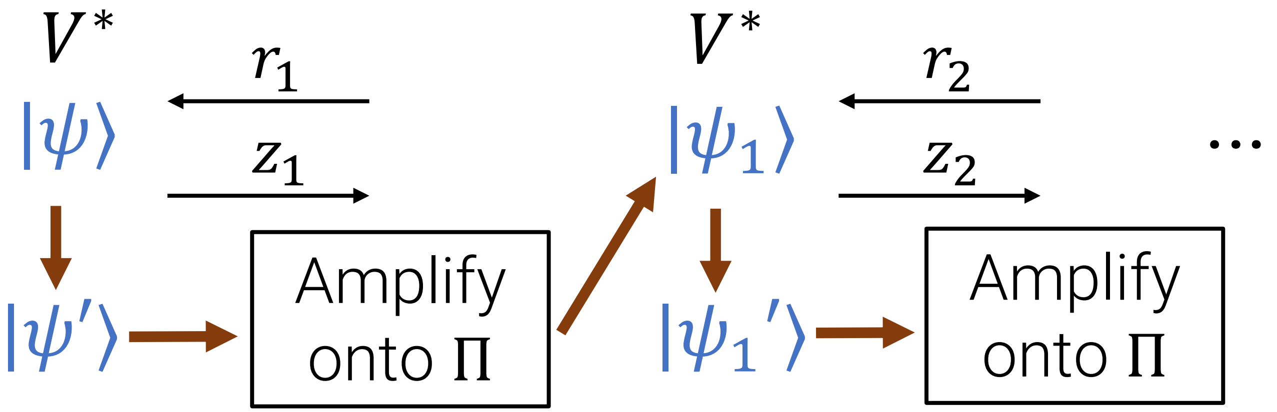
1) Run U_{Ext} (coherent version of Ext)

2) Extract b from $(r,z),(r',z')$

3) Run $U_{\mathrm{Ext}}^{\dagger}$

[CMSZ21] allows us to extract b, but will **disturb** $|\psi\rangle$.

CMSZ rewinding: define projector Π onto "useful" verifier states.



V^*

$|\psi\rangle$

r_1

z_1

$|\psi'\rangle$

Amplify onto Π

V^*

$|\psi_1\rangle$

r_2

z_2

$|\psi_1{}'\rangle$

Amplify onto Π

…

Idea: define variable-time procedure

$\mathrm{Ext} =$"Run CMSZ to get valid $(r, z), (r', z')$"

Simulator:

1) Run U_{Ext} (coherent version of Ext)

2) Extract b from $(r, z), (r', z')$

3) Run $U_{\mathrm{Ext}}^{\dagger}$

This works if:

- Ext is measured-EQPT
- Ext succeeds with probability ≈ 1
- b is unique.

Big problem: CMSZ-based $\mathbf{Ext}$ doesn’t run in expected poly-time!

Big problem: CMSZ-based Ext doesn't run in expected poly-time!

If V^* gives valid z with prob ε, it takes $1/\varepsilon$ expected loops of "measure + repair" to get another accepting response.

Big problem: CMSZ-based $\mathbf{Ext}$ doesn't run in expected poly-time!

If V^* gives valid z with prob ε, it takes $1/\varepsilon$ expected loops of "measure + repair" to get another accepting response.

$$\text{Expected runtime is } \varepsilon \underbrace{\left(\frac{1}{\varepsilon}\right)}_{\text{Classical runtime}} \underset{\text{CMSZ repair runtime}}{\left(\frac{1}{\varepsilon^4}\right)} = \mathbf{unbounded}$$

Big problem: CMSZ-based $\mathbf{Ext}$ doesn't run in expected poly-time!

If V^* gives valid z with prob ε, it takes $1/\varepsilon$ expected loops of "measure + repair" to get another accepting response.

Expected runtime is $\varepsilon\left(\frac{1}{\varepsilon}\right)\left(\frac{1}{\varepsilon^4}\right) = \mathbf{unbounded}$

This work:

- New rewinding template where accepting (r, z) is generated by amplifying adversary onto *guaranteed accepting executions* instead of querying on random r.

Big problem: CMSZ-based $\mathbf{Ext}$ doesn't run in expected poly-time!

If V^* gives valid z with prob ε, it takes $1/\varepsilon$ expected loops of "measure + repair" to get another accepting response.

$$\text{Expected runtime is } \varepsilon\left(\frac{1}{\varepsilon}\right)\left(\frac{1}{\varepsilon^4}\right) = \mathbf{unbounded}$$

This work:

- New rewinding template where accepting (r, z) is generated by amplifying adversary onto *guaranteed accepting executions* instead of querying on random r.
  - Not obvious: why does repair still work?

Big problem: CMSZ-based $\mathbf{Ext}$ doesn't run in expected poly-time!

If V^* gives valid z with prob ε, it takes $1/\varepsilon$ expected loops of "measure + repair" to get another accepting response.

$$\text{Expected runtime is } \varepsilon\left(\frac{1}{\varepsilon}\right)\left(\frac{1}{\varepsilon^4}\right) = \mathbf{unbounded}$$

This work:

- New rewinding template where accepting (r, z) is generated by amplifying adversary onto *guaranteed accepting executions* instead of querying on random r.
  - Not obvious: why does repair still work?
- Further speed up CMSZ-style repair with new, variable-time algorithms based on the quantum singular value transform [GSLW19].

Conclusions

- Definitions and reductions for post-quantum ZK are subtle!
- We define post-quantum ZK using **coherent** EQPT simulation.
- We build coherent EQPT simulators for textbook ZK protocols by combining a new rewinding template with variable-time algorithms based on [CMSZ21,GLSW19].

Conclusions

- Definitions and reductions for post-quantum ZK are subtle!
- We define post-quantum ZK using **coherent** EQPT simulation.
- We build coherent EQPT simulators for textbook ZK protocols by combining a new rewinding template with variable-time algorithms based on [CMSZ21,GLSW19].

Future directions

- Develop general theory of post-quantum ZK.
- Quantum-communication ZK? Natural starting point: [BG20, GJMZ22]

Thanks for listening!